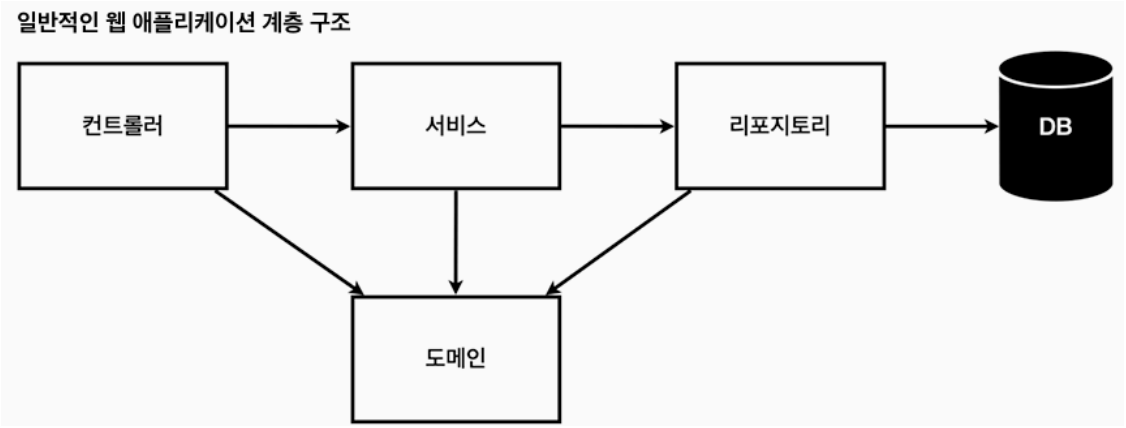
2주차

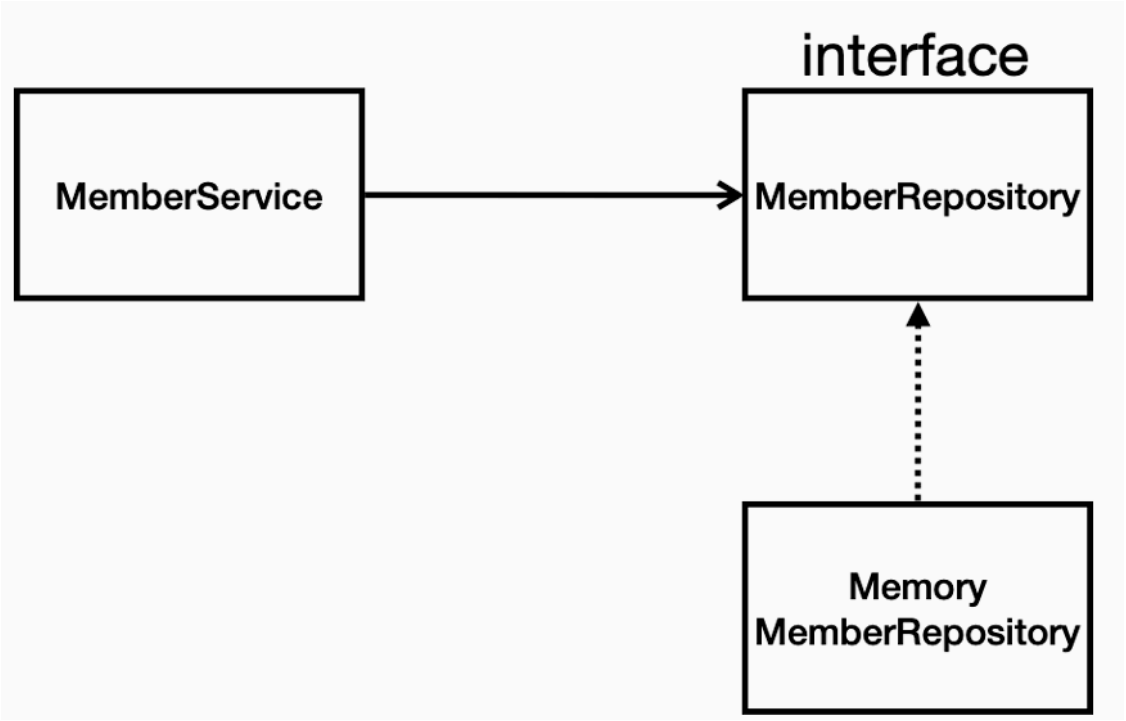
스프링-입문-스프링부트

섹션4. 회원 관리 예제 - 백엔드 개발

비즈니스 요구사항 정리

- 데이터 : 회원 ID, 이름
- 기능 : 회원 등록, 조회
- 데이터 저장소가 선정되지 않음





회원 도메인과 리포지토리 만들기

domain/Member

```
package hello.hello_spring.domain;

public class Member {
    private Long id; //임의의 값 - 데이터 구분을 위한 시스템이 지정하는 값
    private String name;

    public Long getId(){
        return id;
    }

    public void setId(Long id){
        this.id = id;
    }

    public String getName(){
        return name;
    }

    public void setName(String name){
        this.name = name;
    }
}
```

repository/MeberRepository interface

```
package hello.hello_spring.repository;

import hello.hello_spring.domain.Member;

import java.util.List;
import java.util.Optional;

public interface MemberRepository {
    Member save(Member member); //저장
    Optional<Member> findById(Long id); //id로 회원 찾기
```

```
    Optional<Member> findByName(String name); //Optional - 없으면 NULL을 김
    List<Member> findAll();
}
```

repository/MemoryMemberRepository

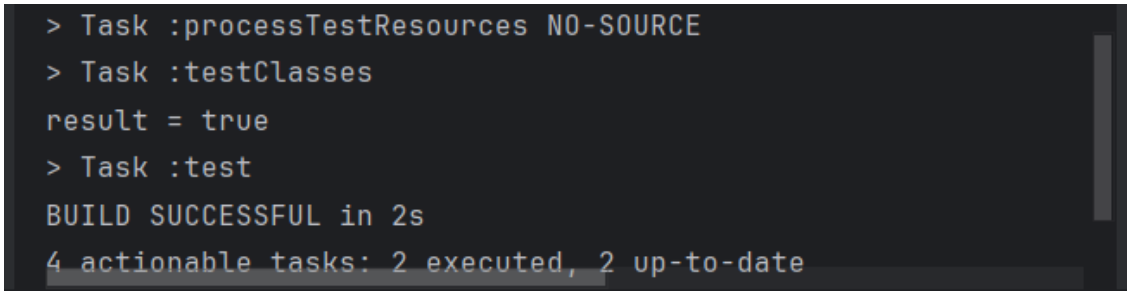
```
package hello.hello_spring.repository;

import hello.hello_spring.domain.Member;

import java.util.*;

public class MemoryMemberRepository implements MemberRepository {

    private static Map<Long, Member> store = new HashMap<>(); //저장
    private static long sequence = 0L;

    @Override
    public Member save(Member member) { //Member save
        member.setId(++sequence);  //member id값 세팅
        store.put(member.getId(), member); //store에 저장 -> map에 저장
        return null;
    }

    @Override
    public Optional<Member> findById(Long id) {
        return Optional.ofNullable(store.get(id)); //sotre에서 꺼내기
        // 결과가 없을경우 NULL (Optional NUllable)
    }

    @Override
    public List<Member> findAll() {
        return new ArrayList<>(store.values());
    }

    @Override
    public Optional<Member> findByName(String name) {
        return store.values().stream()
```

```
            .filter(member -> member.getName().equals(name)) //가져온것과 para
            .findAny();
    }
}
```

어떻게 검증을 할까? → Testcase 작성

회원 레포지토리 테스트 케이스 작성

```
package hello.hello_spring.repository;

import hello.hello_spring.domain.Member;
import org.assertj.core.api.Assertions;
import org.junit.jupiter.api.Test;

class MemoryMemberRepositoryTest {

    MemoryMemberRepository repository = new MemoryMemberRepository();

    @Test
    public void save(){
        Member member = new Member();
        member.setName("spring");

        repository.save(member);

        Member result = repository.findById(member.getId()).get();
        Assertions.assertThat(member).isEqualTo(result);
        //Assertions.assertEquals(member, result);
        //System.out.println("result = " + (result == member));
    }
}
```



⇒ 글자를 실제로 볼 수 없다.

```
Assertions.assertEquals(member, result);
```

```
Assertions.assertThat(member).isEqualTo(result);
```

```
    @Test
    public void findByName() {
        Member member1 = new Member();
        member1.setName("spring1");
        repository.save(member1);

        Member member2 = new Member();
        member1.setName("spring1");
        repository.save(member2);

        Optional<Member> result = Optional.of(repository.findByName("spring1"
    }
}
```

spring1을 조회하는 메서드

```
    @Test
    public void findAll() {
        Member member1 = new Member();
        member1.setName("spring1");
        repository.save(member1);
```

```
        Member member2 = new Member();
        member1.setName("spring1");
        repository.save(member2);

        List<Member> result = repository.findAll();

        Assertions.assertThat(result.size()).isEqualTo(2);
    }
```

총 2개가 저장되었는지 확인

```
public void clearStore(){
        store.clear();
    }

    //테스트는 순서와 관계없이 작동되기 때문에 초기화를 해줘야한다.
```

```
@AfterEach
    void afterEach() {
        repository.clearStore();
    }
```

회원 서비스 개발

```
package hello.hello_spring.sevice;

import hello.hello_spring.domain.Member;
import hello.hello_spring.repository.MemberRepository;
import hello.hello_spring.repository.MemoryMemberRepository;

import java.util.List;
import java.util.Optional;

public class MemberService {
```

```
    private final MemberRepository memberRepository = new MemoryMember
    //회원가입
    public Long join(Member member) {
        //같은 이름이 있는 중복 회원 X
        validateDuplicateMember(member);
        memberRepository.save(member);
        return member.getId();
    }
    private void validateDuplicateMember(Member member) {
        memberRepository.findByName(member.getName())
                .ifPresent(m->{
                    throw new IllegalArgumentException("이미 존재하는회원입니다.");
                });
    }
    //전체 회원 조회
    public List<Member> findMembers() {
        return memberRepository.findAll();
    }

    public Optional<Member> findOne(Long memberId) {
        return memberRepository.findById(memberId);
    }
}
```

회원 서비스 테스트

//given → 무언가 주어졌을때//when → 실행했을때//then → 이런 결과가 나와야한다

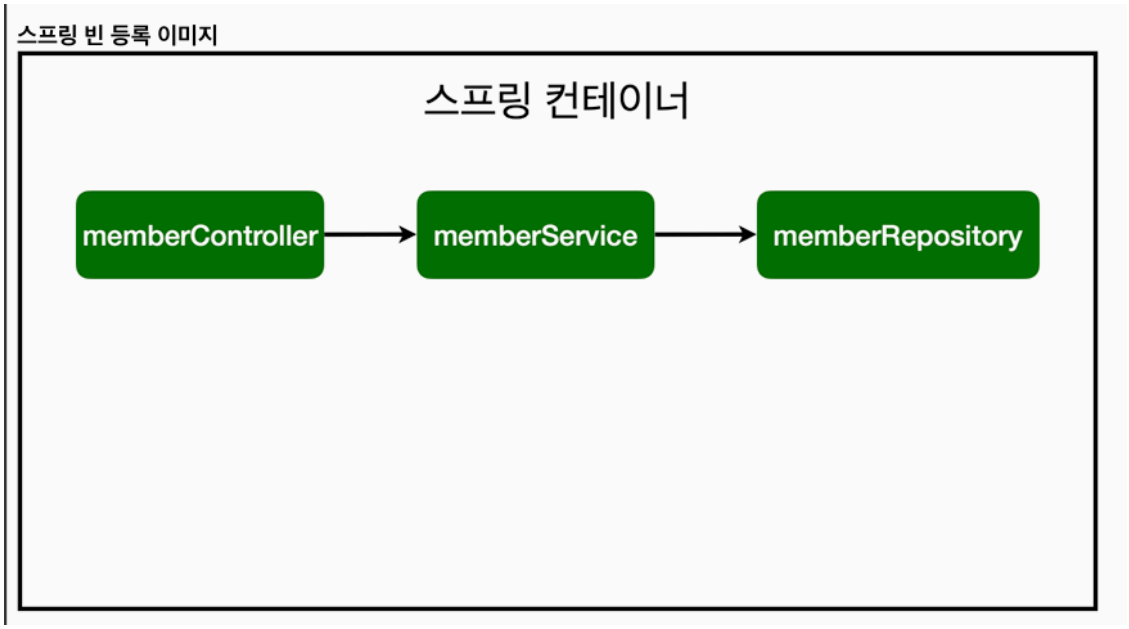
```
class MemberServiceTest {

    MemberService memberService = new MemberService();
    @Test
    void 회원가입(){
        //given → 무언가 주어졌을때
        Member member = new Member();
        member.setName("hello");
        //when → 실행했을때
```

```
        Long saveId = memberService.join(member);
        //then → 이런 결과가 나와야한다
        Member findMember = memberService.findOne(saveId).get();
        Assertions.assertThat(member.getId()).isEqualTo(findMember.getId());
    }
```

섹션5. 스프링빈과 의존관계

회원 컨트롤러가 회원 서비스와 리포지토리를 사용할 수 있게 의존관계

회원 컨트롤러에 의존 관계 추가



연결이 필요하다

스프링 빈을 등록하는 방법

- 컴포넌트 스캔과 자동 의존관계 설정
- 자바 코드로 직접 스프링 빈 등록하

싱글톤 - 유일하게 하나만 등록해서 공유한다. 같은 스프링 빈이면 모두 같은 인스턴스

자바 코드로 스프링 빈 등록

```
package hello.hellospring;
 import hello.hellospring.repository.MemberRepository;
 import hello.hellospring.repository.MemoryMemberRepository;
 import hello.hellospring.service.MemberService;
 import org.springframework.context.annotation.Bean;
 import org.springframework.context.annotation.Configuration;
 @Configuration
 public class SpringConfig {
   @Bean
 public MemberService memberService() {
 return new MemberService(memberRepository());
   }
   @Bean
 public MemberRepository memberRepository() {
 return new MemoryMemberRepository();
   }
 }
 ``
```

장점 :

섹션6. 웹 MVC 개발

회원 웹 기능 - 홈 화면 추가

```
package hello.hello_spring.controller;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {

    @GetMapping("/") //그냥 들어오면 home 호출
    public String home() {
        return "home";
```

```
    }
}
```

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<body>
<div class="container">
    <div>
        <h1>Hello Spring</h1>
        <p>회원 기능</p>
        <p>
            <a
                href="/members/new">회원 가입</a> //회원가입으로 이동
            <a
                href="/members">회원 목록</a>//회원 목록으로 이동
        </p>
    </div>
</div> <!-- /container -->
</body>
</html>
```

회원 웹 기능 - 등록

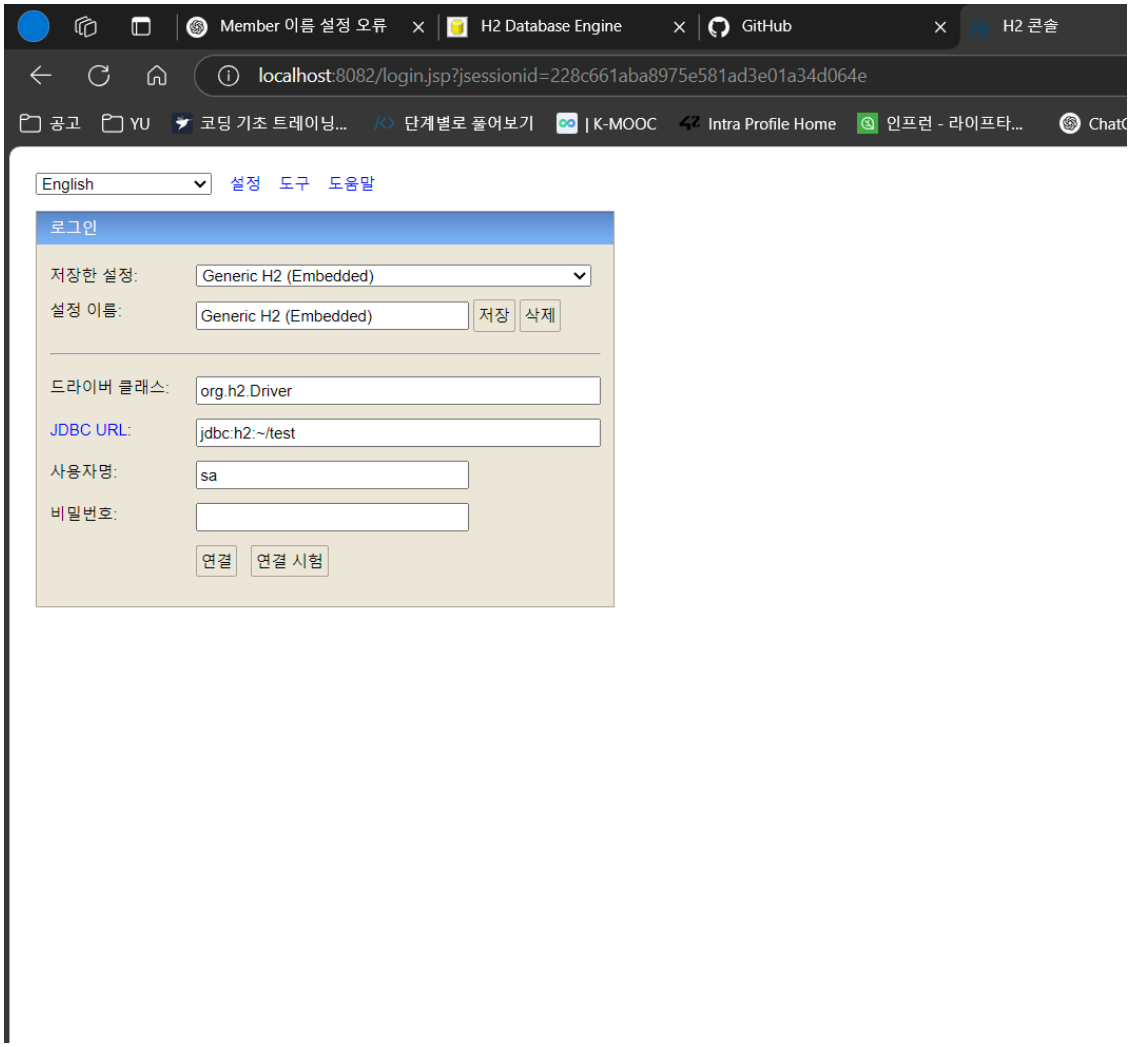
```
package hello.hello_spring.controller;

public class MemberForm {
    private String name;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

회원 웹 기능 - 조회

```
@GetMapping(value = "/members")
public String list(Model model) {
List<Member> members = memberService.findMembers();
  model.addAttribute("members", members);
return "members/memberList";
}
```

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<body>

<div class="container">
 <div>
  <table>
   <thead>
   <tr>
    <th>#</th>
    <th>이름</th>
   </tr>
   </thead>
   <body>
   <tr th:each="member : ${members}"> //모델 안의 값을 꺼낸다.
    <tr th:text="${member.id}"></td>
    <td th:text="${member.name}"></td>
   </tr>
   </body>
  </table>
 </div>
</div> <!-- /container -->
</body>
</html>
```

섹션7.



테이블 만들기

```
create table member
 (
    id
bigint generated by default as identity,
    name varchar(255),
 primary key (id)
 );
```

```
create table member
(
    id bigint generated by default as identity, //db가 자동으로 값을 채워줌.
    name varchar(255),
```

```
    primary key (id)
);
```

h2

```
insert into member(name) values('spring2')
//MEMBER에 들어가면 ID와 NAME을 볼 수 있다.
```

JPA

- SQL쿼리도 자동으로 생성 해준다.
- SQL과 데이터 중심의 설계에서 객체 중심의 설계로 패러다임 전환.
- 개발 생산성을 높일 수 있다.

JPA를 사용하기 위해서 build.gradle에 추가해야한다.

```
// implementation 'org.springframework.boot:spring-boot-starter-jdbc'
 implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
```

JPA 설정 추가

```
 package hello.hellospring.repository;
 import hello.hellospring.domain.Member;
 import javax.persistence.EntityManager;
 import java.util.List;
 import java.util.Optional;
 public class JpaMemberRepository implements MemberRepository {
 private final EntityManager em;
 public JpaMemberRepository(EntityManager em) {
 this.em = em;
   }
 public Member save(Member member) {
```

```
        em.persist(member);
 return member;
   }
 public Optional<Member> findById(Long id) {
 Member member = em.find(Member.class, id);
 return Optional.ofNullable(member);
   }
 public List<Member> findAll() {
 return em.createQuery("select m from Member m", Member.class)
             .getResultList();
   }
public Optional<Member> findByName(String name) {
 List<Member> result = em.createQuery("select m from Member m where
m.name = :name", Member.class)
             .setParameter("name", name)
             .getResultList();
 return result.stream().findAny();
   }
 }
```

스프링 데이터 JPA

JPA란?

Java Persistence API : 자바 객체를 데이터베이스에 매핑하는 ORM기술

⇒ JPA를 편리하게 사용하도록 도와주는 기술

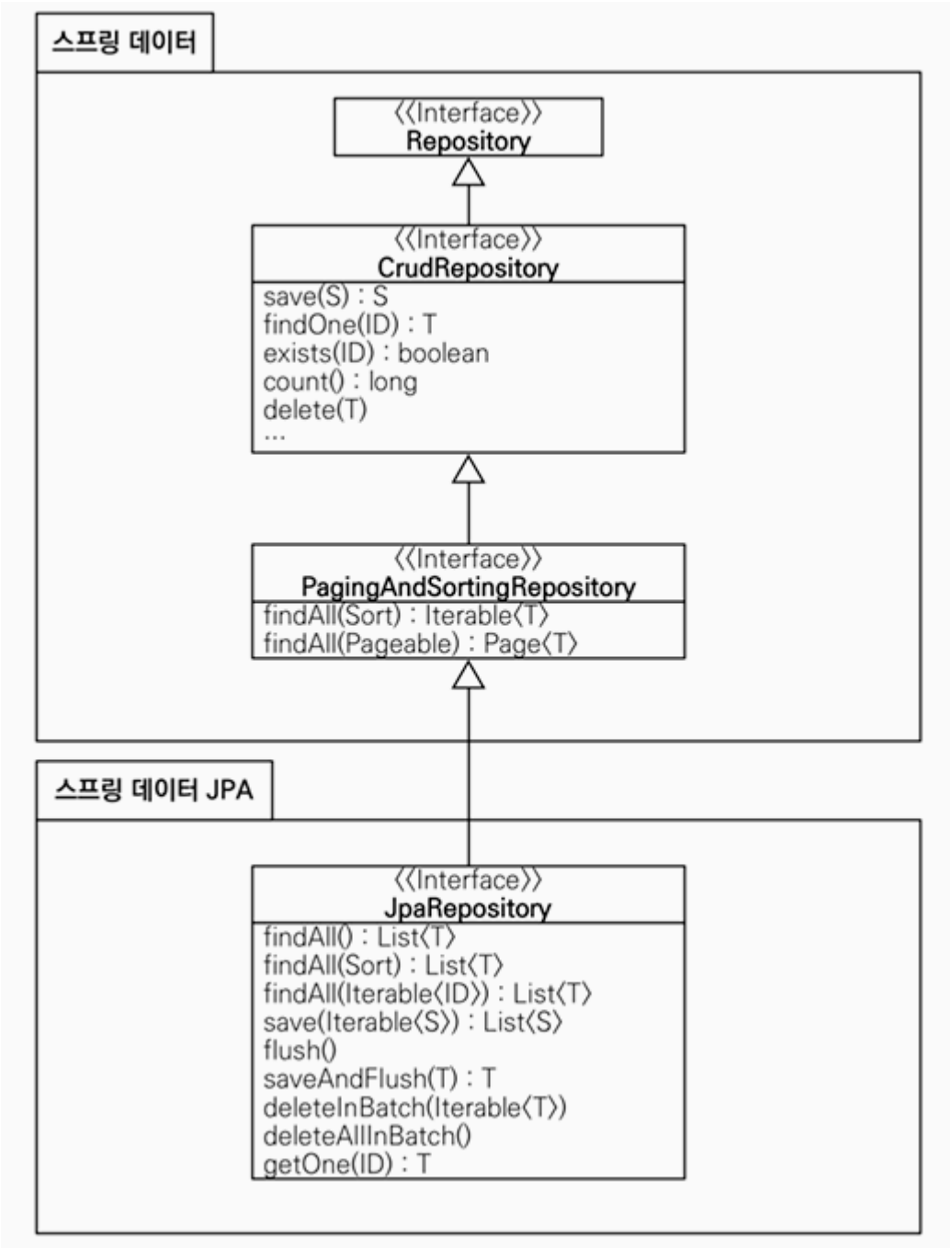
스프링 데이터 JPA회원 리포지토리

```
package hello.hello_spring.repository;

import hello.hello_spring.domain.Member;

import java.util.Optional;

public interface SpringDataJpaMemberRepository extends JpaRepository<Me

    @Override
```

```
    Optional<Member> findByName(String name);
}
```

스프링 설정 변경

```

```



스프링 DB 접근 기술

- 순수 Jdbc → 각 쿼리가 어마어마하다
- 스프링 통합 테스트
- 스프링 JdbcTemplate
- JPA → 쿼리를 작성할 필요가 없다
- 스플이 데이터 JPA → 인터페이스로만으로 작성 가능